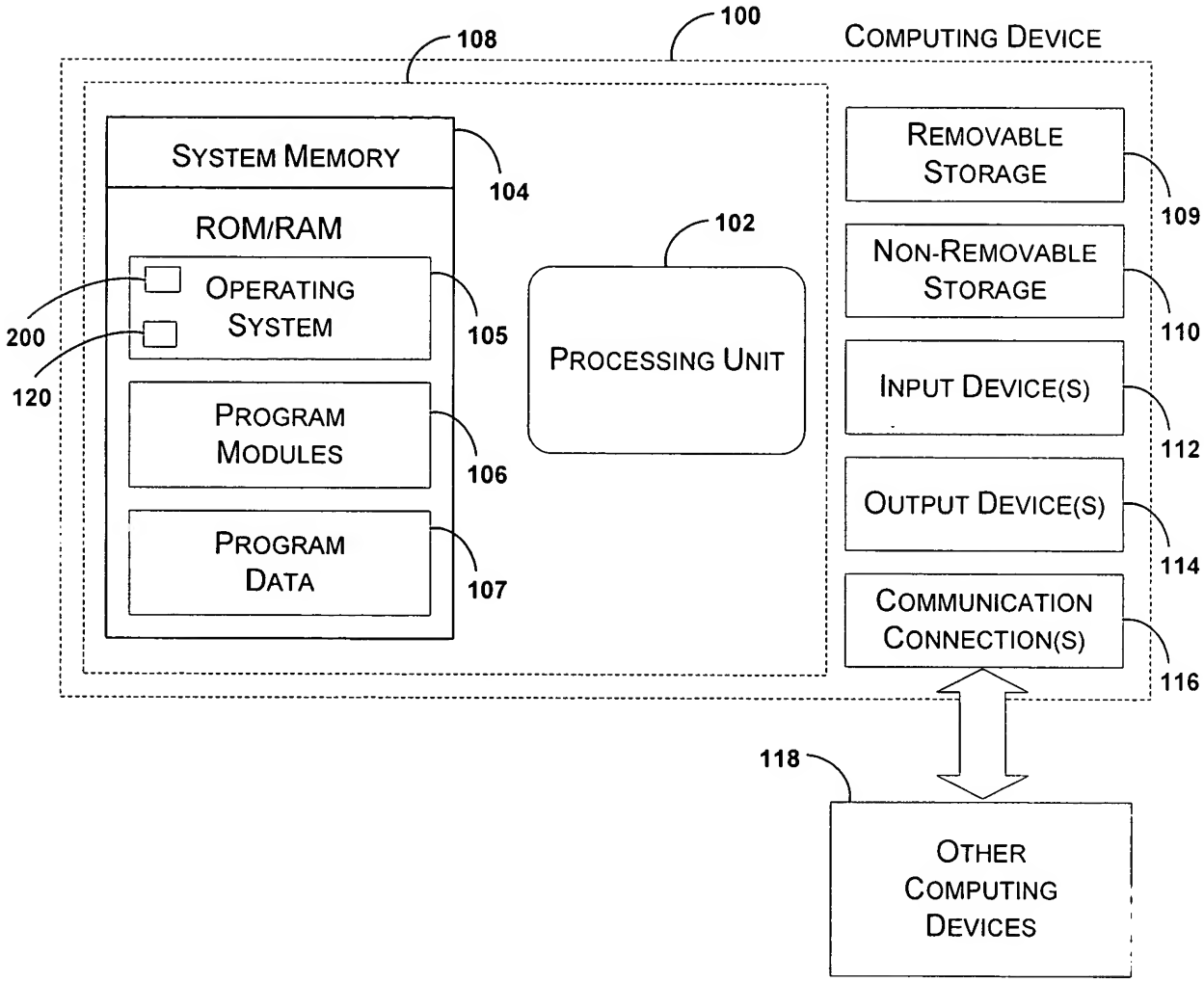
100
COMPUTING DEVICE
108
SYSTEM MEMORY
104
ROM/RAM
OPERATING SYSTEM
105
200
120
PROGRAM MODULES
106
PROGRAM DATA
107
102
PROCESSING UNIT
REMOVABLE STORAGE
109
NON-REMOVABLE STORAGE
110
INPUT DEVICE(S)
112
OUTPUT DEVICE(S)
114
COMMUNICATION CONNECTION(S)
116
118
OTHER COMPUTING DEVICES

Fig. 1

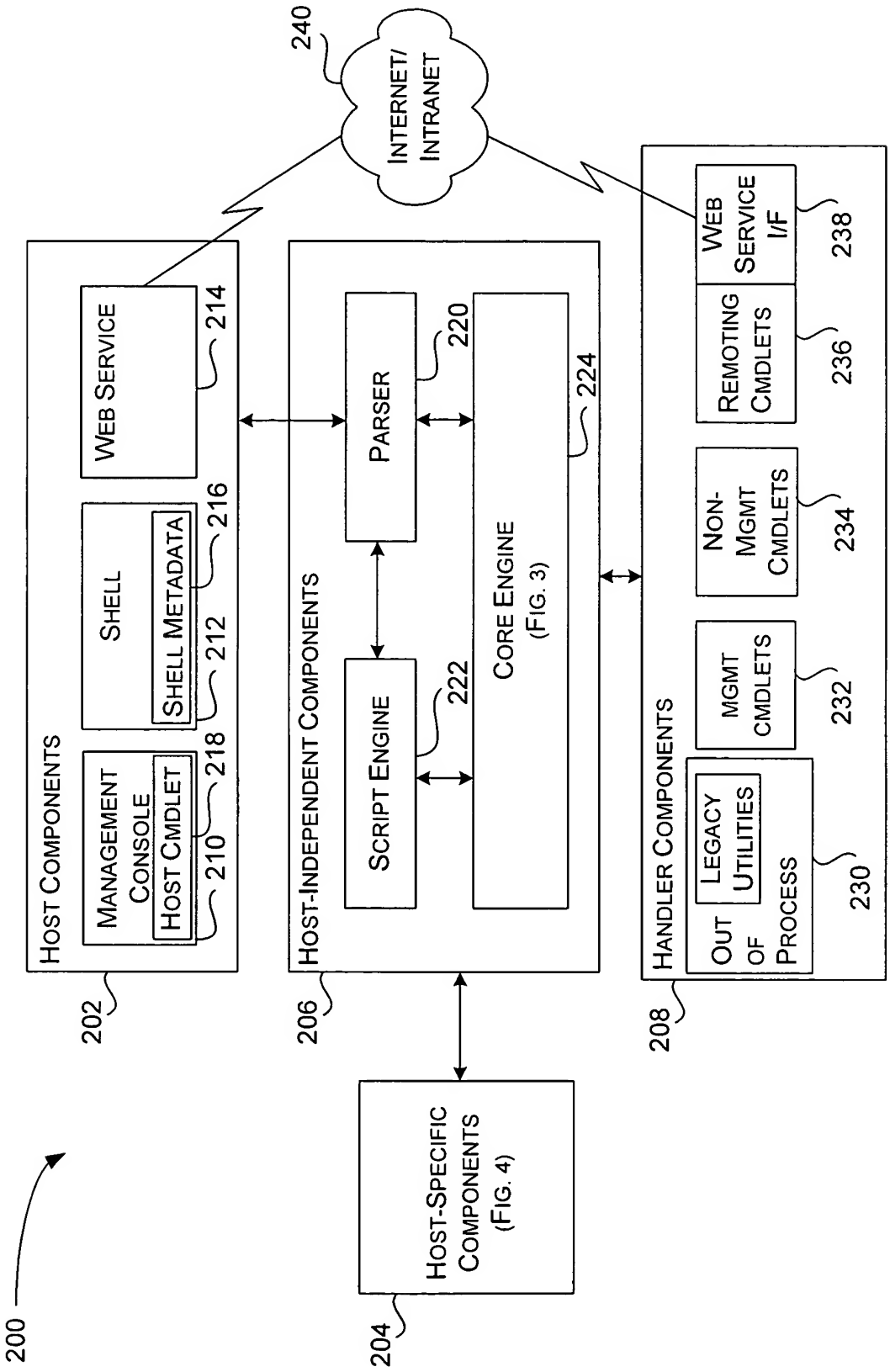
200
240
INTERNET/ INTRANET
HOST COMPONENTS
202
MANAGEMENT CONSOLE
HOST CMDLET
210
218
SHELL
SHELL METADATA
212
216
WEB SERVICE
214
HOST-INDEPENDENT COMPONENTS
206
SCRIPT ENGINE
222
PARSER
220
CORE ENGINE (FIG. 3)
224
HANDLER COMPONENTS
208
OUT OF PROCESS
LEGACY UTILITIES
230
MGMT CMDLETS
232
NON-MGMT CMDLETS
234
REMOTING CMDLETS
236
WEB SERVICE I/F
238
HOST-SPECIFIC COMPONENTS (FIG. 4)
204

Fig. 2

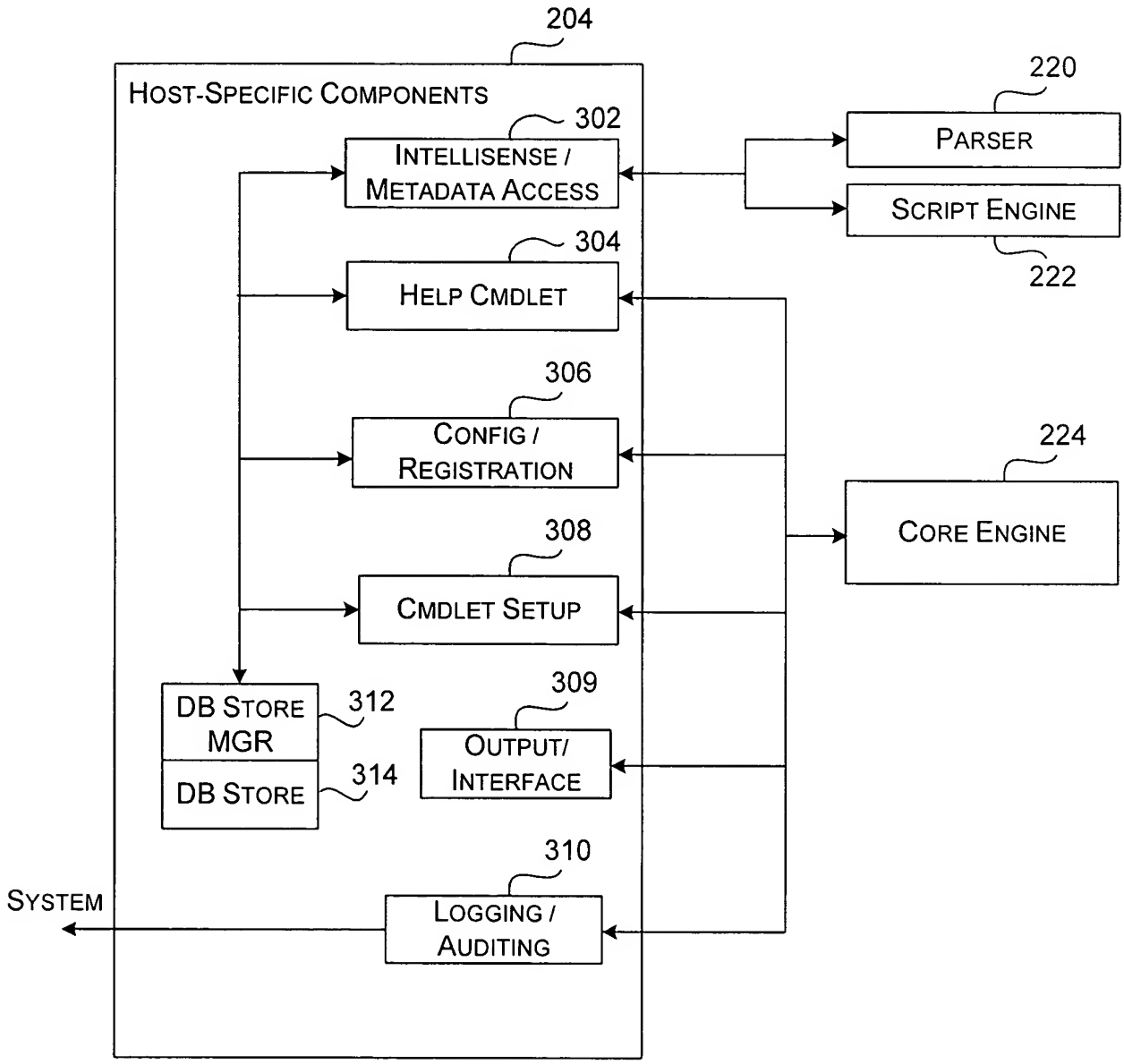
204
HOST-SPECIFIC COMPONENTS
302
INTELLISENSE / METADATA ACCESS
304
HELP CMDLET
306
CONFIG / REGISTRATION
308
CMDLET SETUP
DB STORE MGR
312
DB STORE
314
309
OUTPUT/ INTERFACE
310
LOGGING / AUDITING
SYSTEM
220
PARSER
SCRIPT ENGINE
222
224
CORE ENGINE

Fig. 3

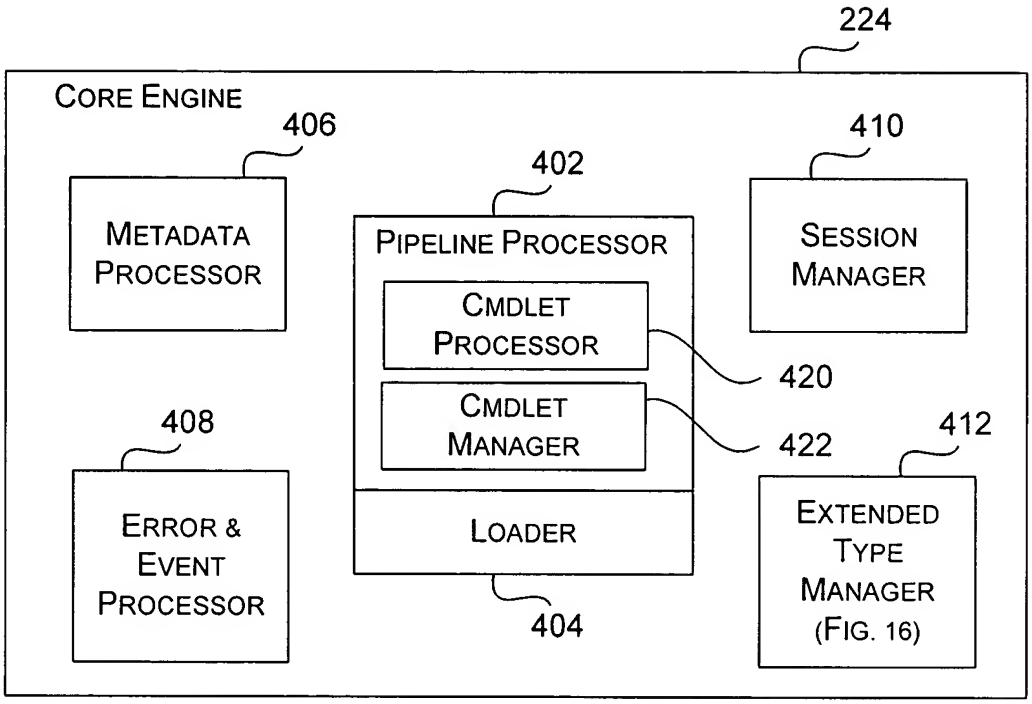
224
CORE ENGINE
406
METADATA PROCESSOR
402
PIPELINE PROCESSOR
CMDLET PROCESSOR
420
CMDLET MANAGER
422
LOADER
404
410
SESSION MANAGER
408
ERROR & EVENT PROCESSOR
412
EXTENDED TYPE MANAGER (FIG. 16)

Fig. 4

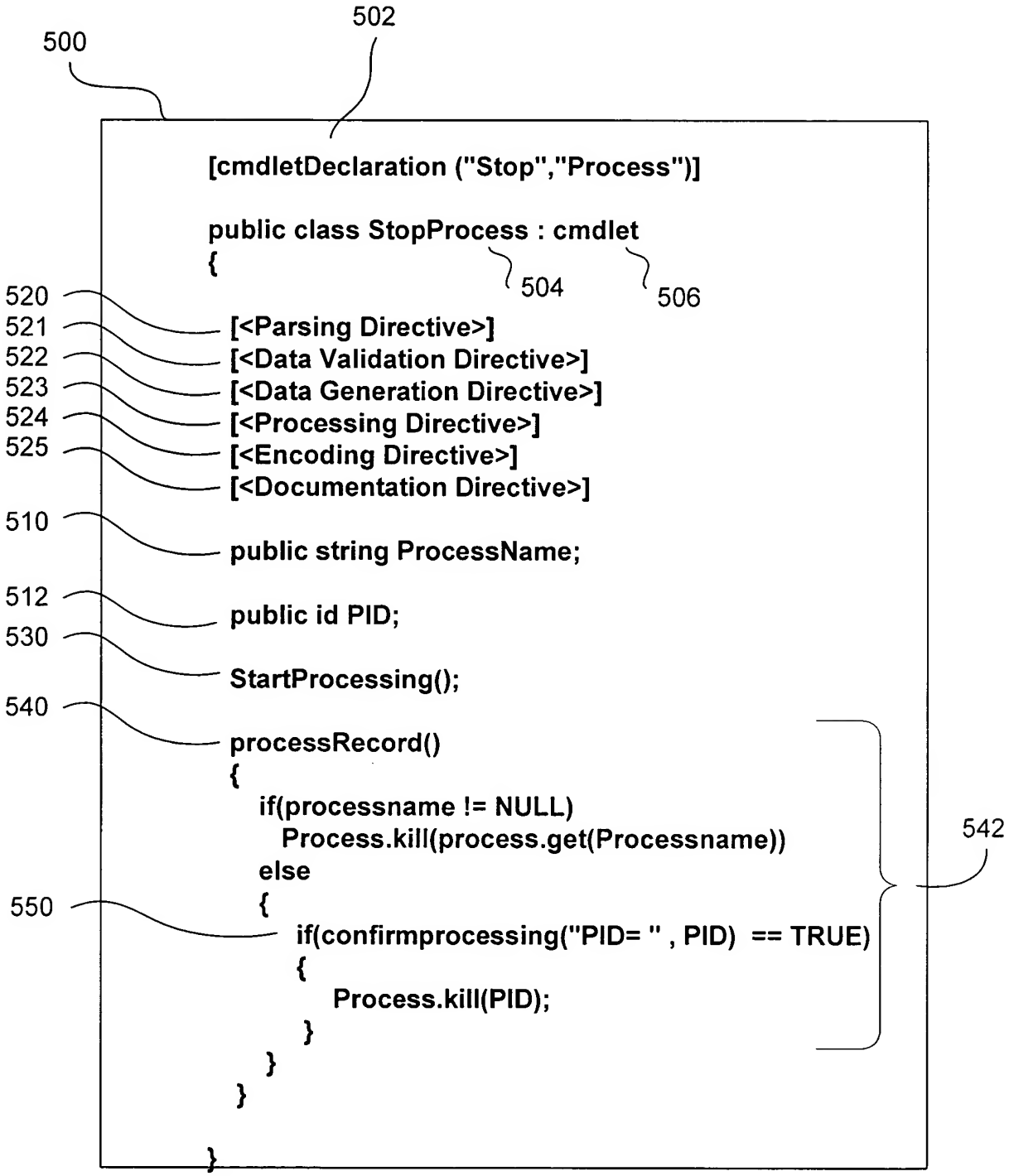
500
502
[cmdletDeclaration ("Stop","Process")]
public class StopProcess : cmdlet
{
504
506
520
521
522
523
524
525
[<Parsing Directive>]
[<Data Validation Directive>]
[<Data Generation Directive>]
[<Processing Directive>]
[<Encoding Directive>]
[<Documentation Directive>]
510
public string ProcessName;
512
public id PID;
530
StartProcessing();
540
processRecord()
{
if(processname != NULL)
Process.kill(process.get(Processname))
else
{
550
if(confirmprocessing("PID= " , PID) == TRUE)
{
Process.kill(PID);
}
}
}
}
542

Fig. 5

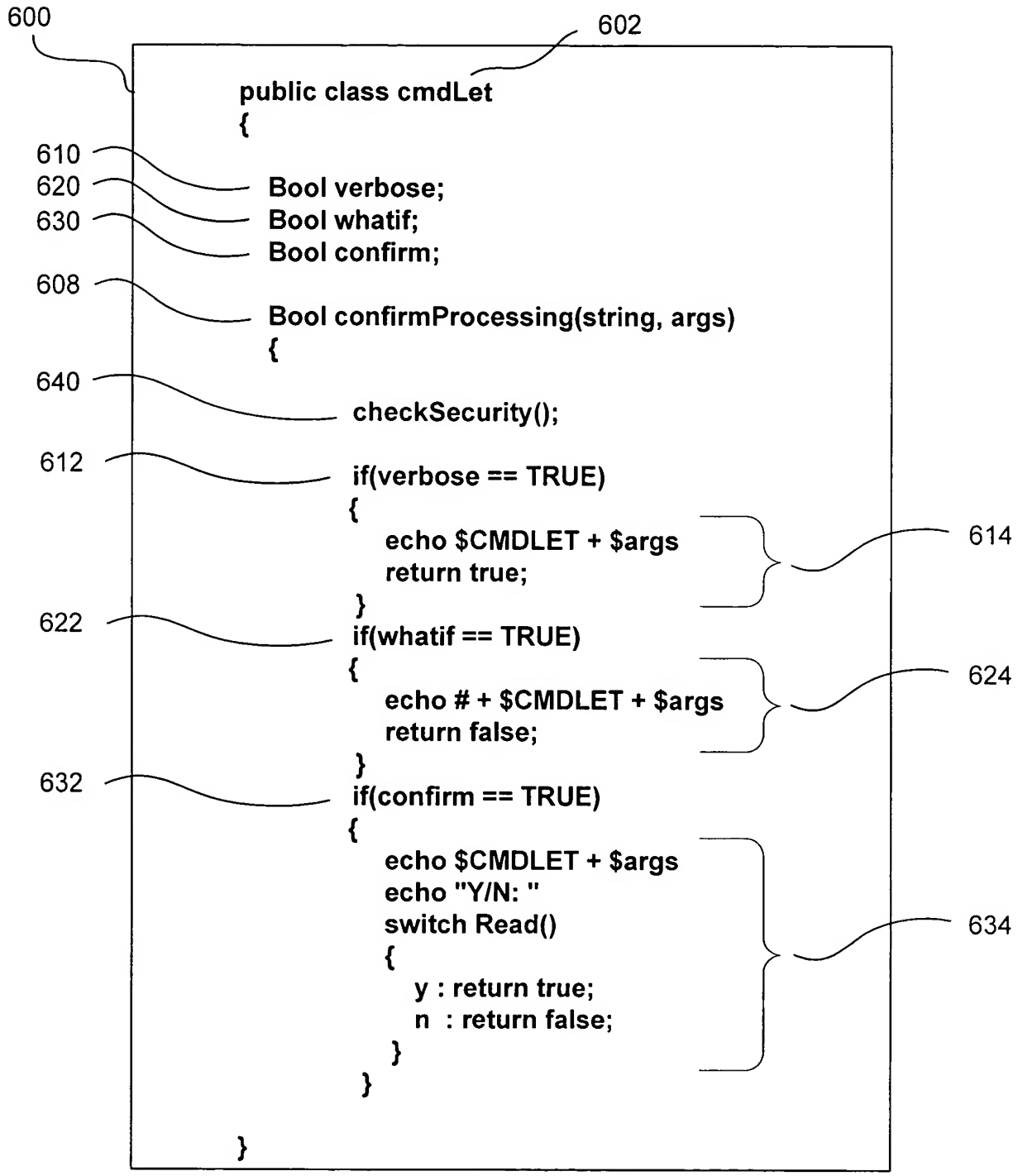
600
602
public class cmdLet
{
610
Bool verbose;
620
Bool whatif;
630
Bool confirm;
608
Bool confirmProcessing(string, args)
{
640
checkSecurity();
612
if(verbose == TRUE)
{
echo $CMDLET + $args
return true;
}
614
622
if(whatif == TRUE)
{
echo # + $CMDLET + $args
return false;
}
624
632
if(confirm == TRUE)
{
echo $CMDLET + $args
echo "Y/N: "
switch Read()
{
y : return true;
n : return false;
}
}
634
}

Fig. 6

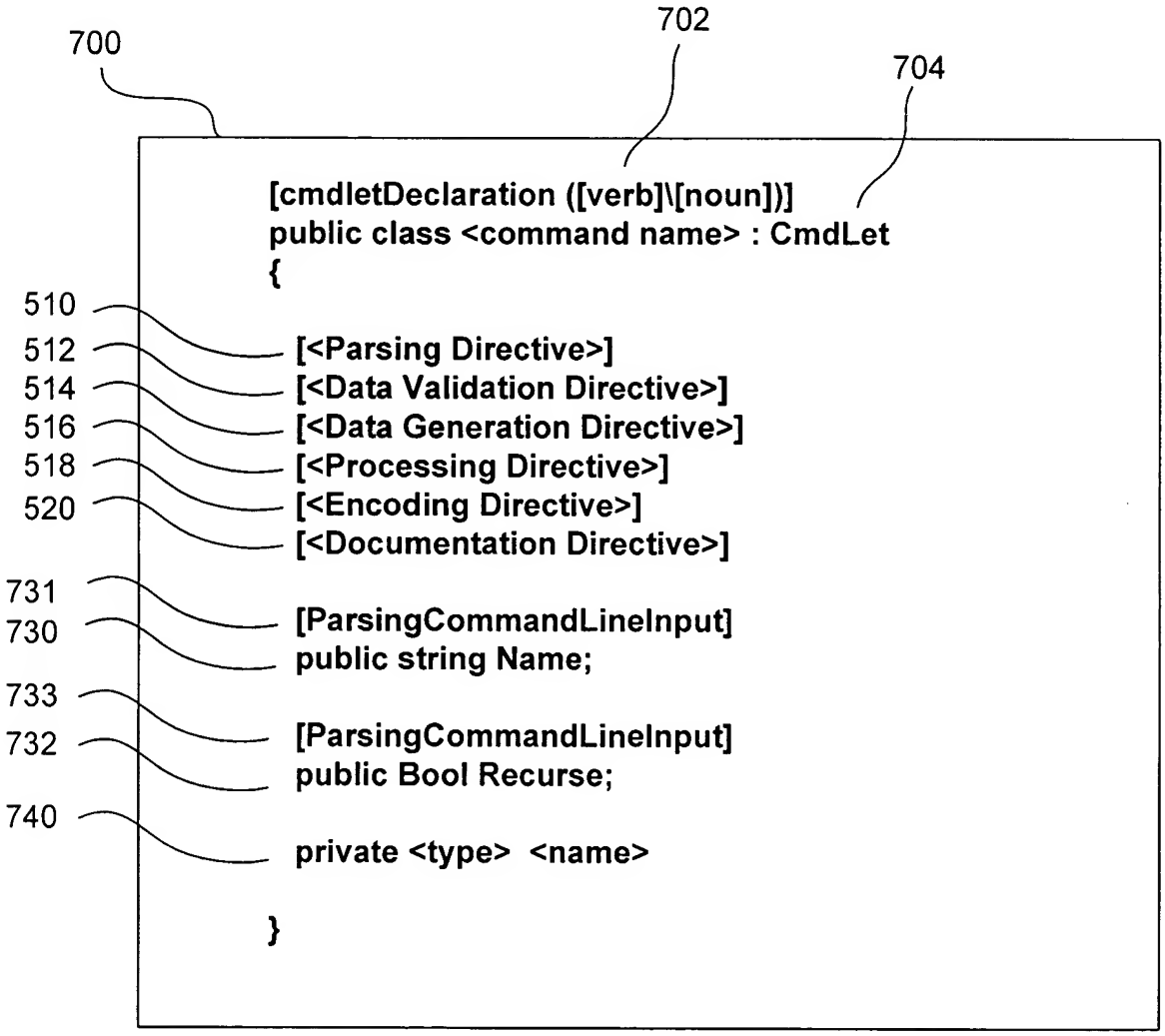
700
702
704
[cmdletDeclaration ([verb]\[noun])]
public class <command name> : CmdLet
{
510
[<Parsing Directive>]
512
[<Data Validation Directive>]
514
[<Data Generation Directive>]
516
[<Processing Directive>]
518
[<Encoding Directive>]
520
[<Documentation Directive>]
731
[ParsingCommandLineInput]
730
public string Name;
733
[ParsingCommandLineInput]
732
public Bool Recurse;
740
private <type> <name>
}

Fig. 7

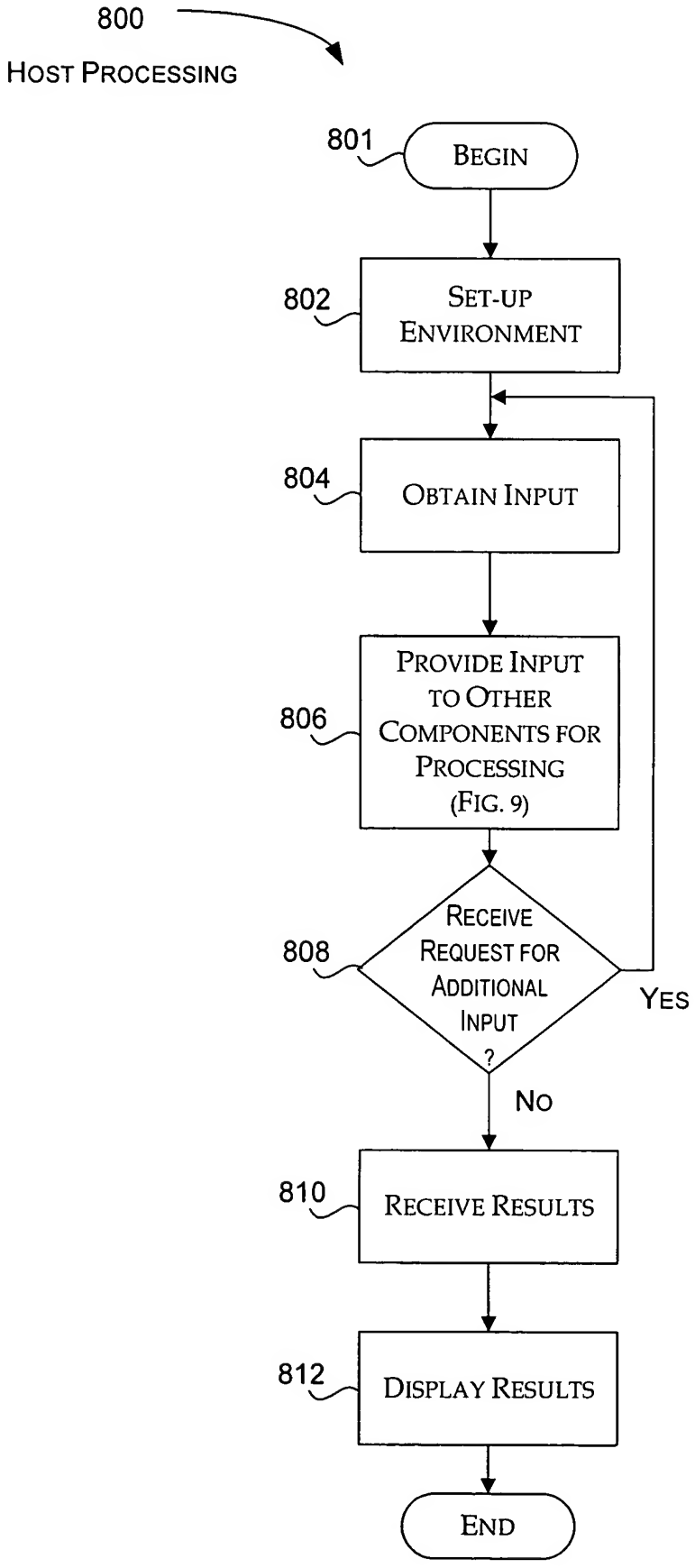
800
HOST PROCESSING
801
BEGIN
802
SET-UP ENVIRONMENT
804
OBTAIN INPUT
806
PROVIDE INPUT TO OTHER COMPONENTS FOR PROCESSING (FIG. 9)
808
RECEIVE REQUEST FOR ADDITIONAL INPUT ?
YES
NO
810
RECEIVE RESULTS
812
DISPLAY RESULTS
END

Fig. 8

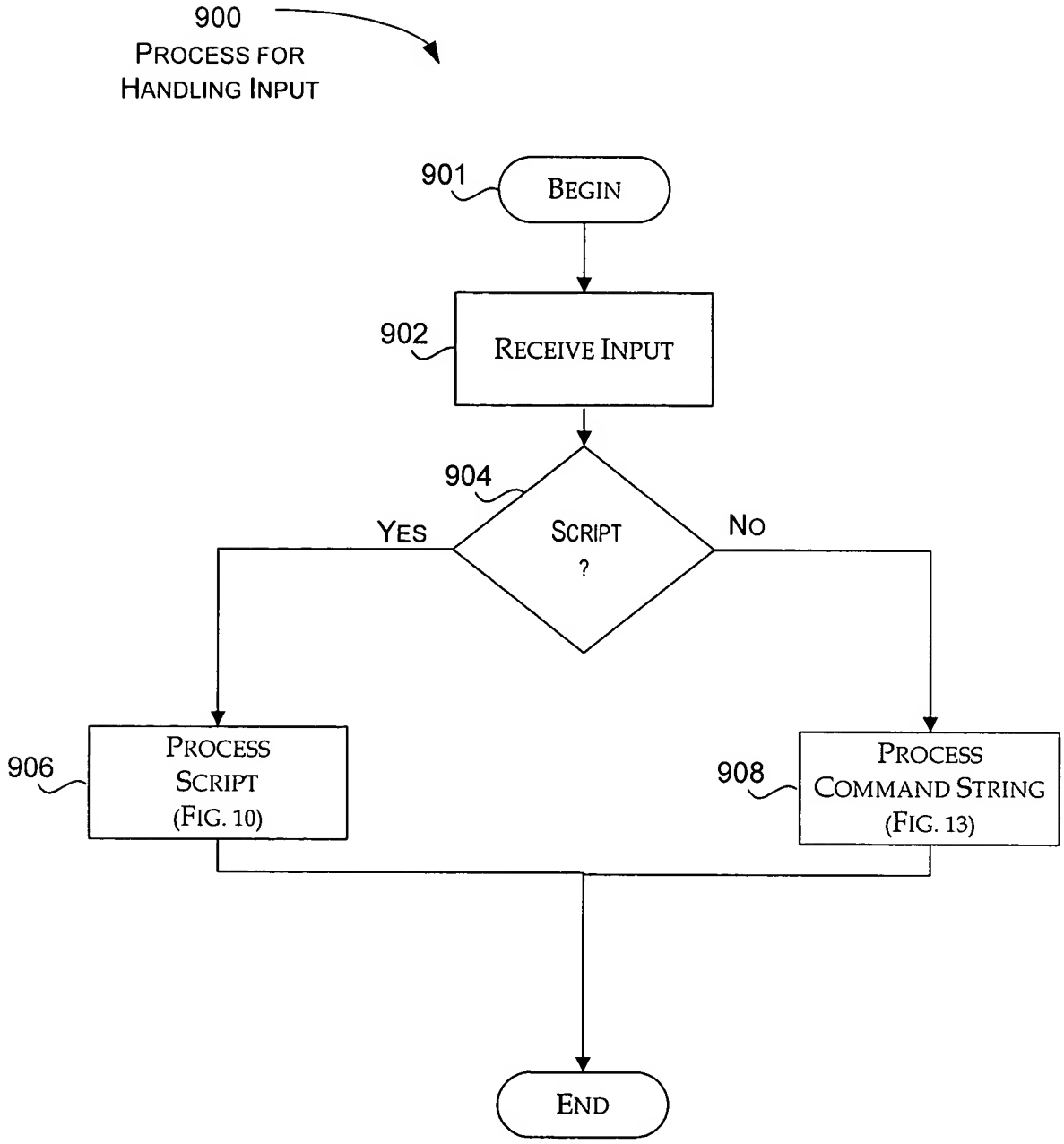
900
PROCESS FOR HANDLING INPUT
901
BEGIN
902
RECEIVE INPUT
904
SCRIPT ?
YES
NO
906
PROCESS SCRIPT (FIG. 10)
908
PROCESS COMMAND STRING (FIG. 13)
END

Fig. 9

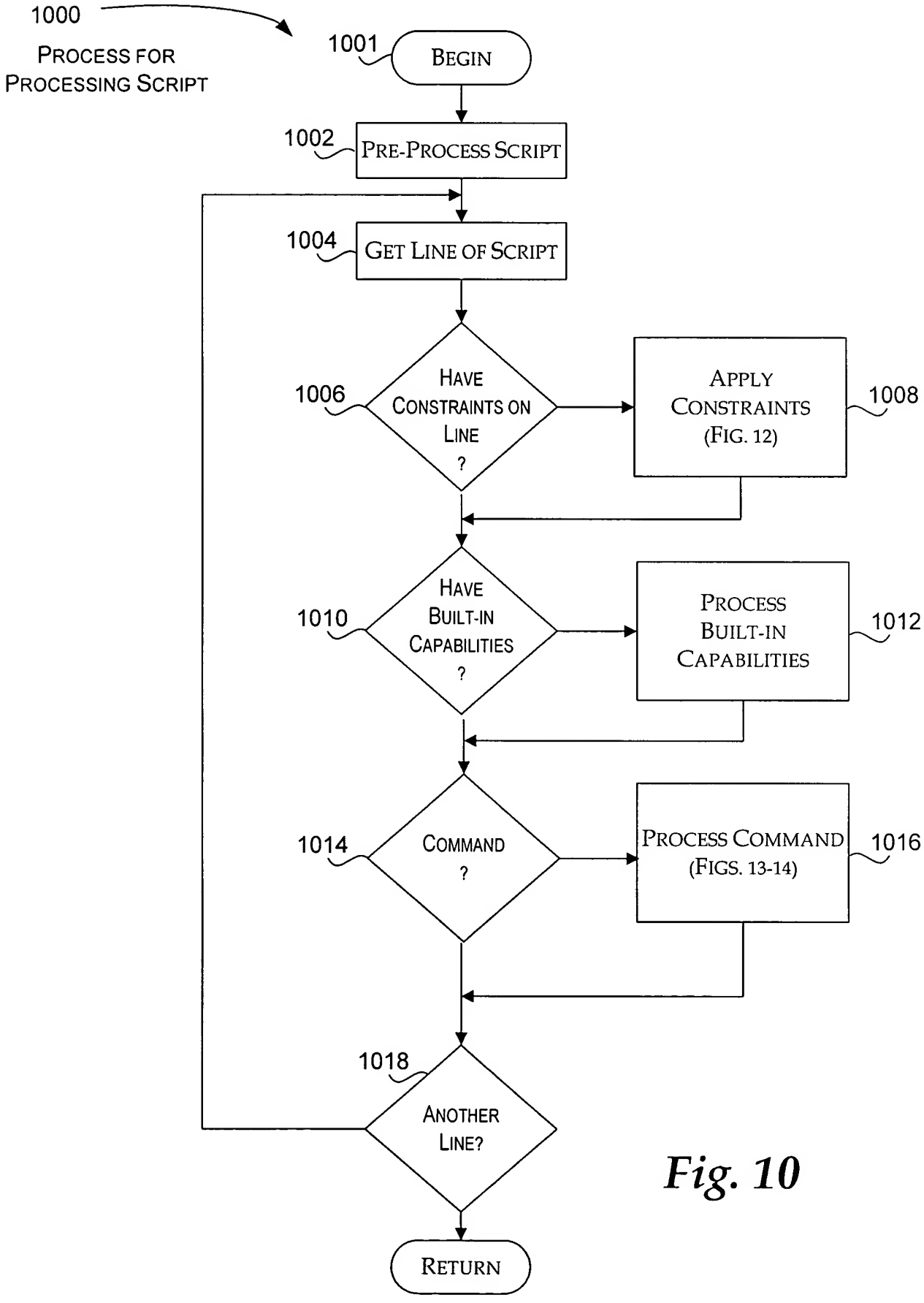
1000
PROCESS FOR PROCESSING SCRIPT
1001
BEGIN
1002
PRE-PROCESS SCRIPT
1004
GET LINE OF SCRIPT
1006
HAVE CONSTRAINTS ON LINE ?
APPLY CONSTRAINTS (FIG. 12)
1008
1010
HAVE BUILT-IN CAPABILITIES ?
PROCESS BUILT-IN CAPABILITIES
1012
1014
COMMAND ?
PROCESS COMMAND (FIGS. 13-14)
1016
1018
ANOTHER LINE?
RETURN

Fig. 10

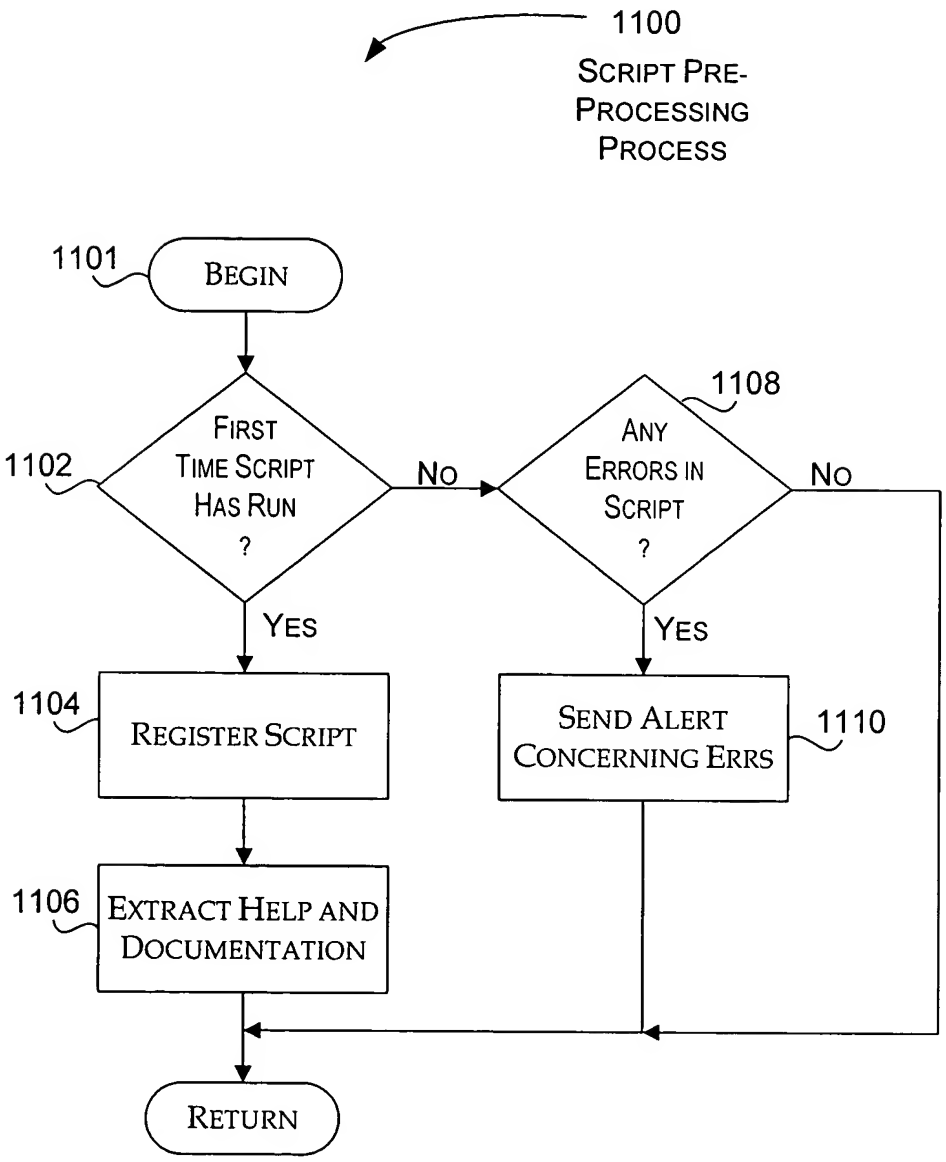
1100
SCRIPT PRE-PROCESSING PROCESS
1101
BEGIN
1102
FIRST TIME SCRIPT HAS RUN ?
NO
1108
ANY ERRORS IN SCRIPT ?
NO
YES
YES
1104
REGISTER SCRIPT
SEND ALERT CONCERNING ERRS
1110
1106
EXTRACT HELP AND DOCUMENTATION
RETURN

Fig. 11

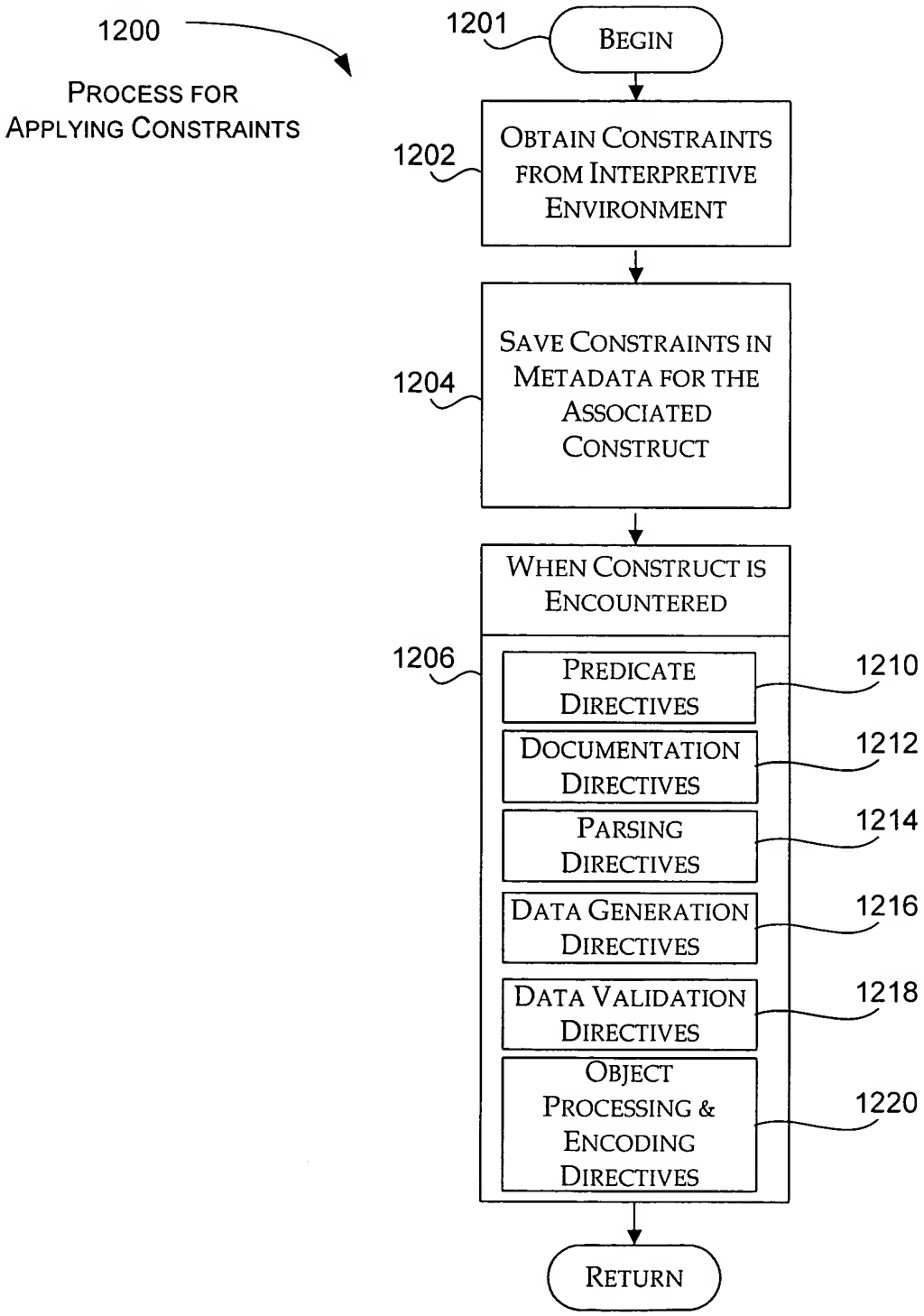
1200
PROCESS FOR APPLYING CONSTRAINTS
1201
BEGIN
1202
OBTAIN CONSTRAINTS FROM INTERPRETIVE ENVIRONMENT
1204
SAVE CONSTRAINTS IN METADATA FOR THE ASSOCIATED CONSTRUCT
WHEN CONSTRUCT IS ENCOUNTERED
1206
PREDICATE DIRECTIVES
1210
DOCUMENTATION DIRECTIVES
1212
PARSING DIRECTIVES
1214
DATA GENERATION DIRECTIVES
1216
DATA VALIDATION DIRECTIVES
1218
OBJECT PROCESSING & ENCODING DIRECTIVES
1220
RETURN

Fig. 12

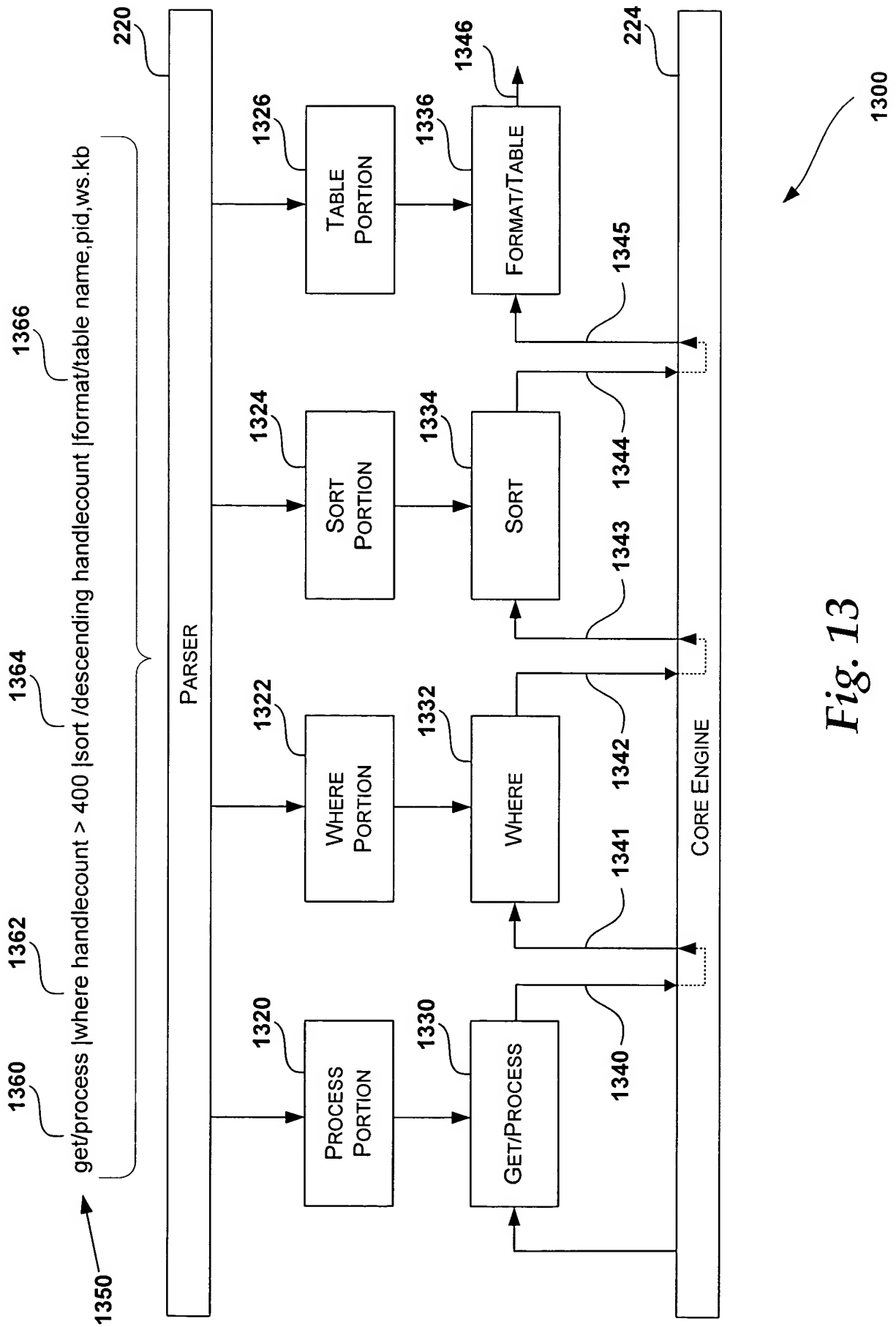
1350
get/process |where handlecount > 400 |sort /descending handlecount |format/table name,pid,ws.kb
1360
1362
1364
1366
PARSER
220
PROCESS PORTION
1320
WHERE PORTION
1322
SORT PORTION
1324
TABLE PORTION
1326
GET/PROCESS
1330
WHERE
1332
SORT
1334
FORMAT/TABLE
1336
CORE ENGINE
224
1340
1341
1342
1343
1344
1345
1346
1300

Fig. 13

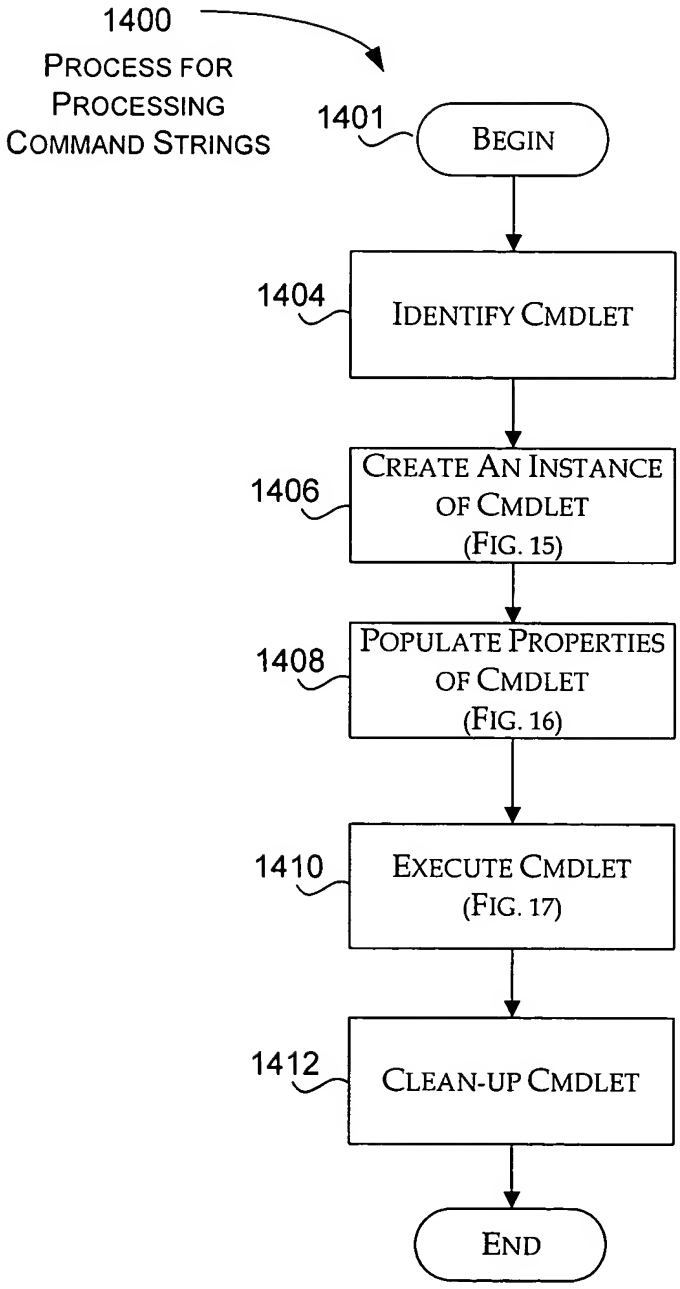
1400
PROCESS FOR PROCESSING COMMAND STRINGS
1401
BEGIN
1404
IDENTIFY CMDLET
1406
CREATE AN INSTANCE OF CMDLET (FIG. 15)
1408
POPULATE PROPERTIES OF CMDLET (FIG. 16)
1410
EXECUTE CMDLET (FIG. 17)
1412
CLEAN-UP CMDLET
END

Fig. 14

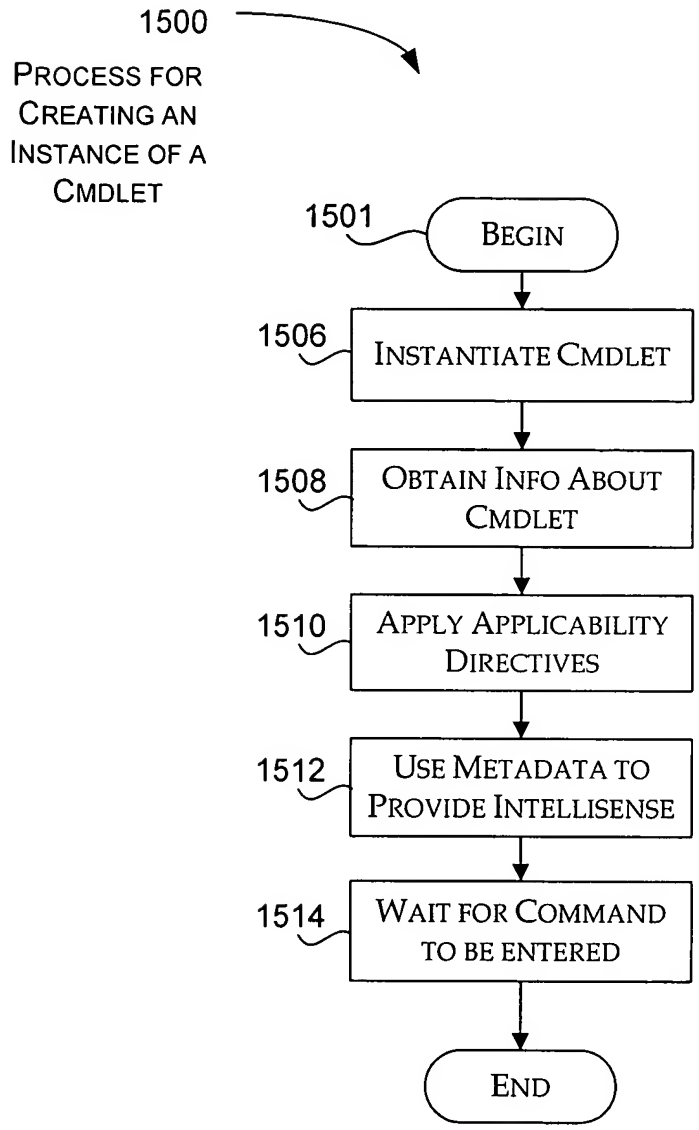
1500
PROCESS FOR CREATING AN INSTANCE OF A CMDLET
1501
BEGIN
1506
INSTANTIATE CMDLET
1508
OBTAIN INFO ABOUT CMDLET
1510
APPLY APPLICABILITY DIRECTIVES
1512
USE METADATA TO PROVIDE INTELLISENSE
1514
WAIT FOR COMMAND TO BE ENTERED
END

Fig. 15

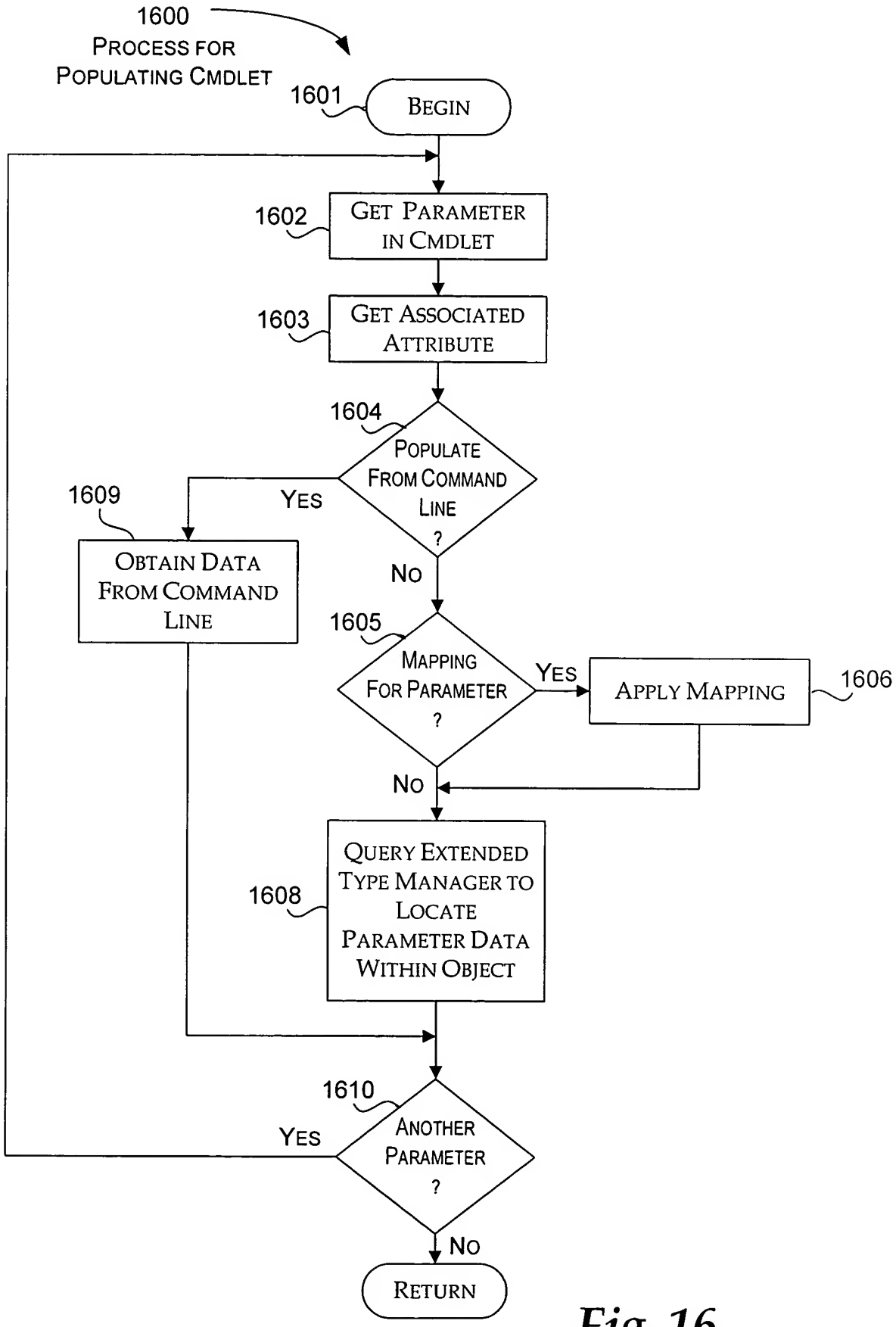
1600
PROCESS FOR POPULATING CMDLET
1601
BEGIN
1602
GET PARAMETER IN CMDLET
1603
GET ASSOCIATED ATTRIBUTE
1604
POPULATE FROM COMMAND LINE ?
YES
1609
OBTAIN DATA FROM COMMAND LINE
NO
1605
MAPPING FOR PARAMETER ?
YES
APPLY MAPPING
1606
NO
1608
QUERY EXTENDED TYPE MANAGER TO LOCATE PARAMETER DATA WITHIN OBJECT
1610
ANOTHER PARAMETER ?
YES
NO
RETURN

Fig. 16

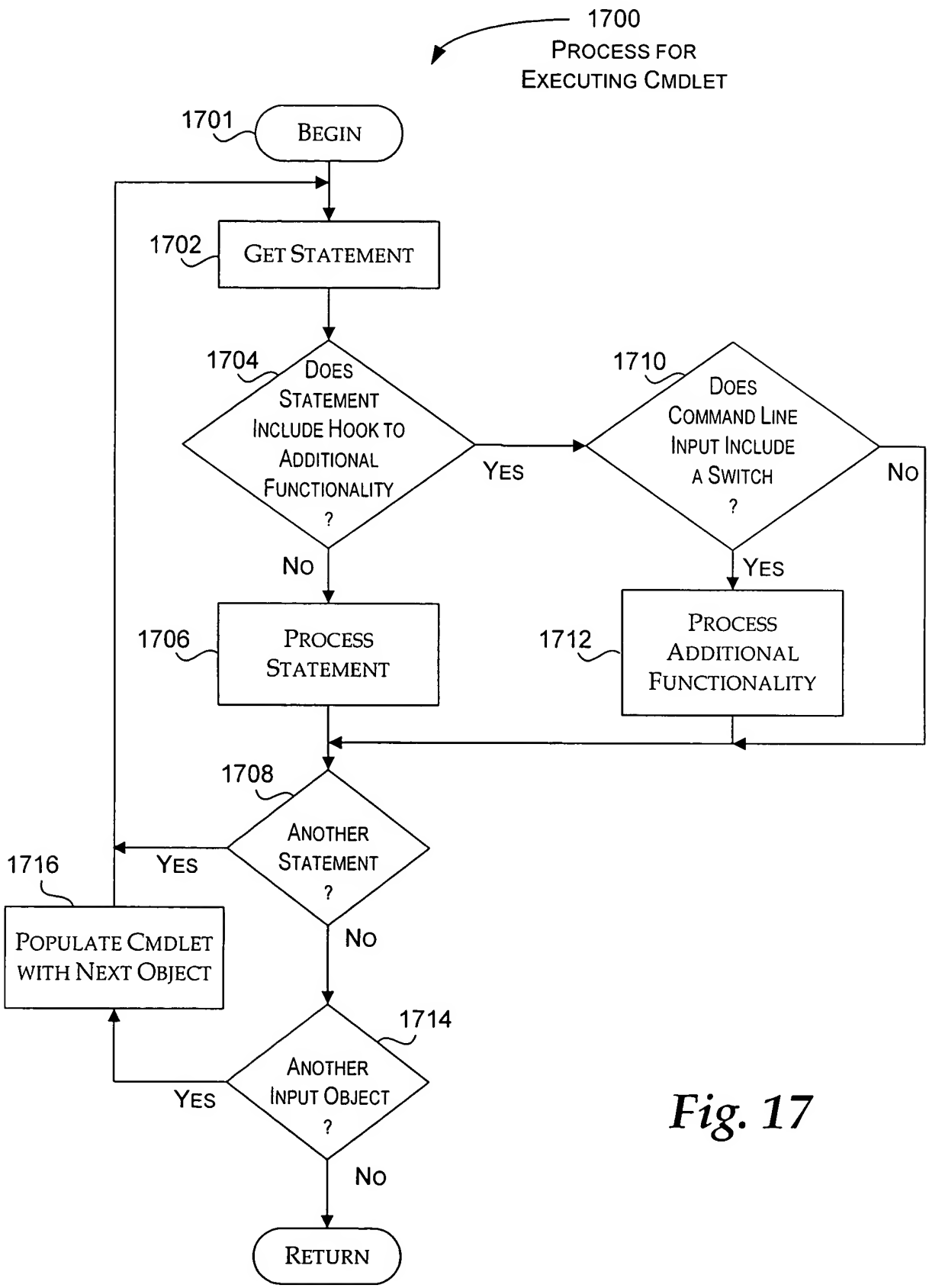
1700
PROCESS FOR EXECUTING CMDLET
1701
BEGIN
1702
GET STATEMENT
1704
DOES STATEMENT INCLUDE HOOK TO ADDITIONAL FUNCTIONALITY ?
YES
NO
1710
DOES COMMAND LINE INPUT INCLUDE A SWITCH ?
YES
NO
1706
PROCESS STATEMENT
1712
PROCESS ADDITIONAL FUNCTIONALITY
1708
ANOTHER STATEMENT ?
YES
NO
1716
POPULATE CMDLET WITH NEXT OBJECT
1714
ANOTHER INPUT OBJECT ?
YES
NO
RETURN

Fig. 17

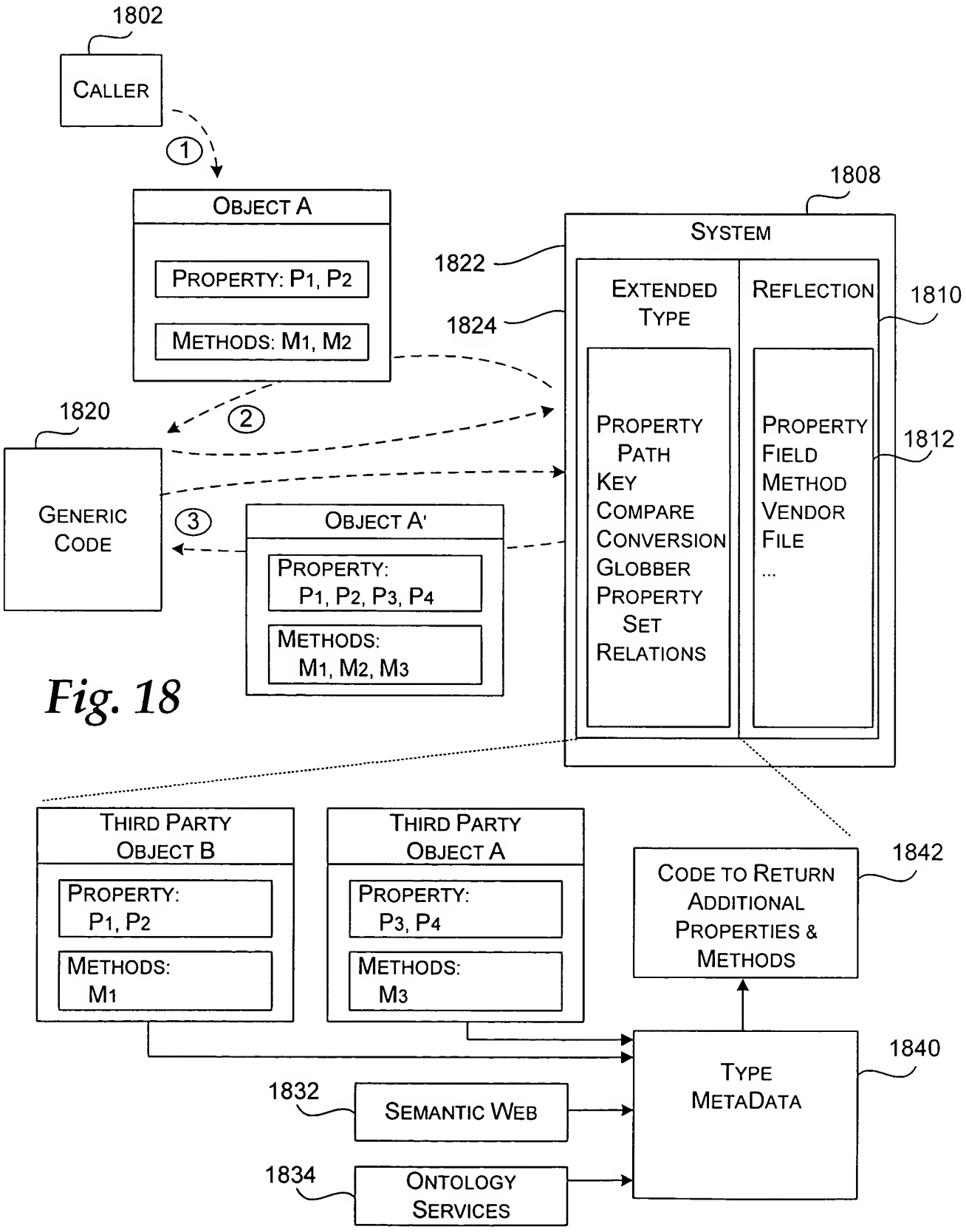
1802
CALLER
1
OBJECT A
PROPERTY: P1, P2
METHODS: M1, M2
1808
SYSTEM
1822
1824
EXTENDED TYPE
REFLECTION
1810
PROPERTY
PATH
KEY
COMPARE
CONVERSION
GLOBBER
PROPERTY
SET
RELATIONS
PROPERTY
FIELD
METHOD
VENDOR
FILE
...
1812
1820
GENERIC CODE
2
3
OBJECT A'
PROPERTY:
P1, P2, P3, P4
METHODS:
M1, M2, M3
THIRD PARTY OBJECT B
PROPERTY:
P1, P2
METHODS:
M1
THIRD PARTY OBJECT A
PROPERTY:
P3, P4
METHODS:
M3
1842
CODE TO RETURN ADDITIONAL PROPERTIES & METHODS
1840
TYPE METADATA
1832
SEMANTIC WEB
1834
ONTOLOGY SERVICES

Fig. 18

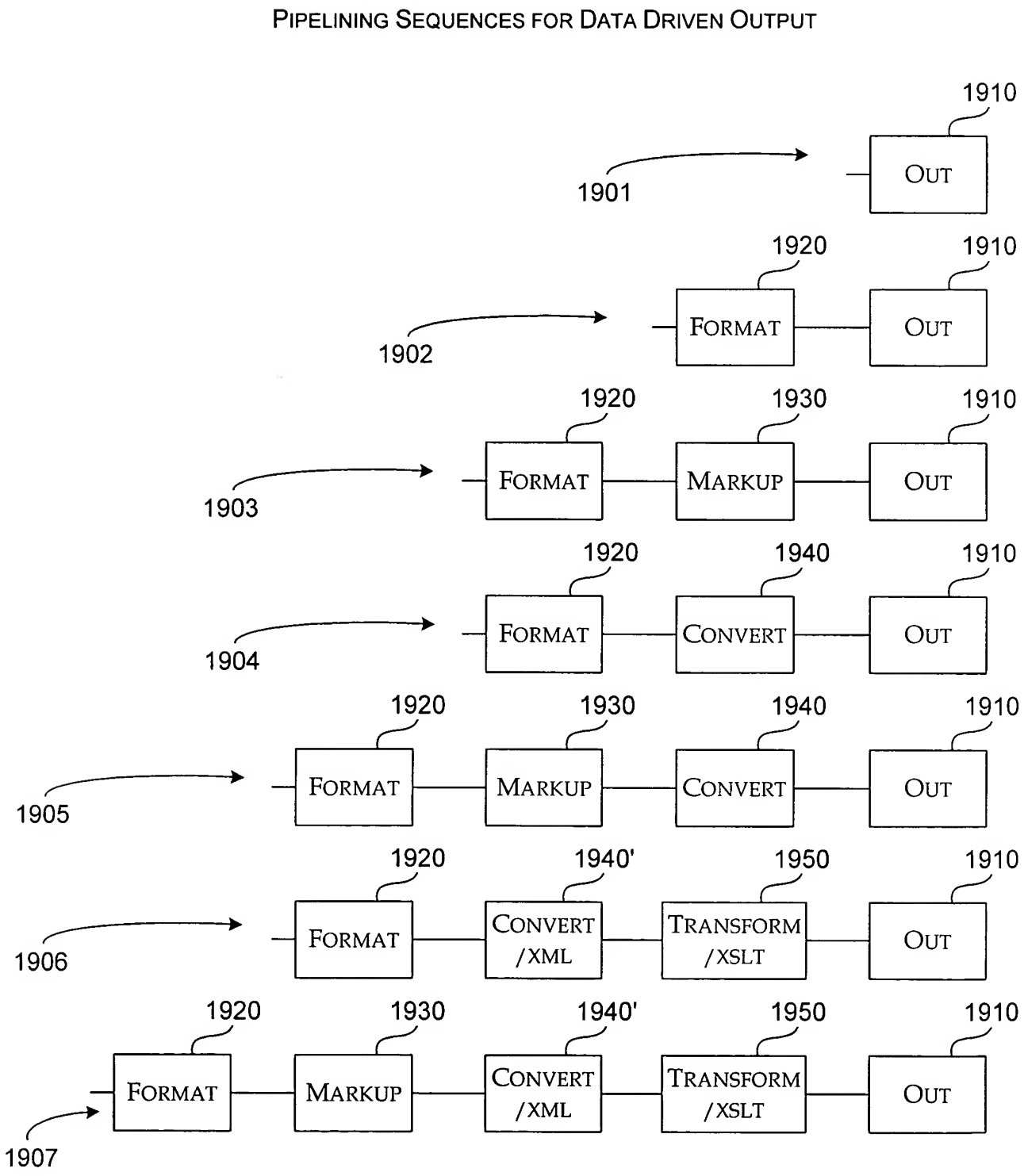
PIPELINING SEQUENCES FOR DATA DRIVEN OUTPUT
1910
1901
OUT
1920
1910
1902
FORMAT
OUT
1920
1930
1910
1903
FORMAT
MARKUP
OUT
1920
1940
1910
1904
FORMAT
CONVERT
OUT
1920
1930
1940
1910
1905
FORMAT
MARKUP
CONVERT
OUT
1920
1940'
1950
1910
1906
FORMAT
CONVERT /XML
TRANSFORM /XSLT
OUT
1920
1930
1940'
1950
1910
1907
FORMAT
MARKUP
CONVERT /XML
TRANSFORM /XSLT
OUT

Fig. 19

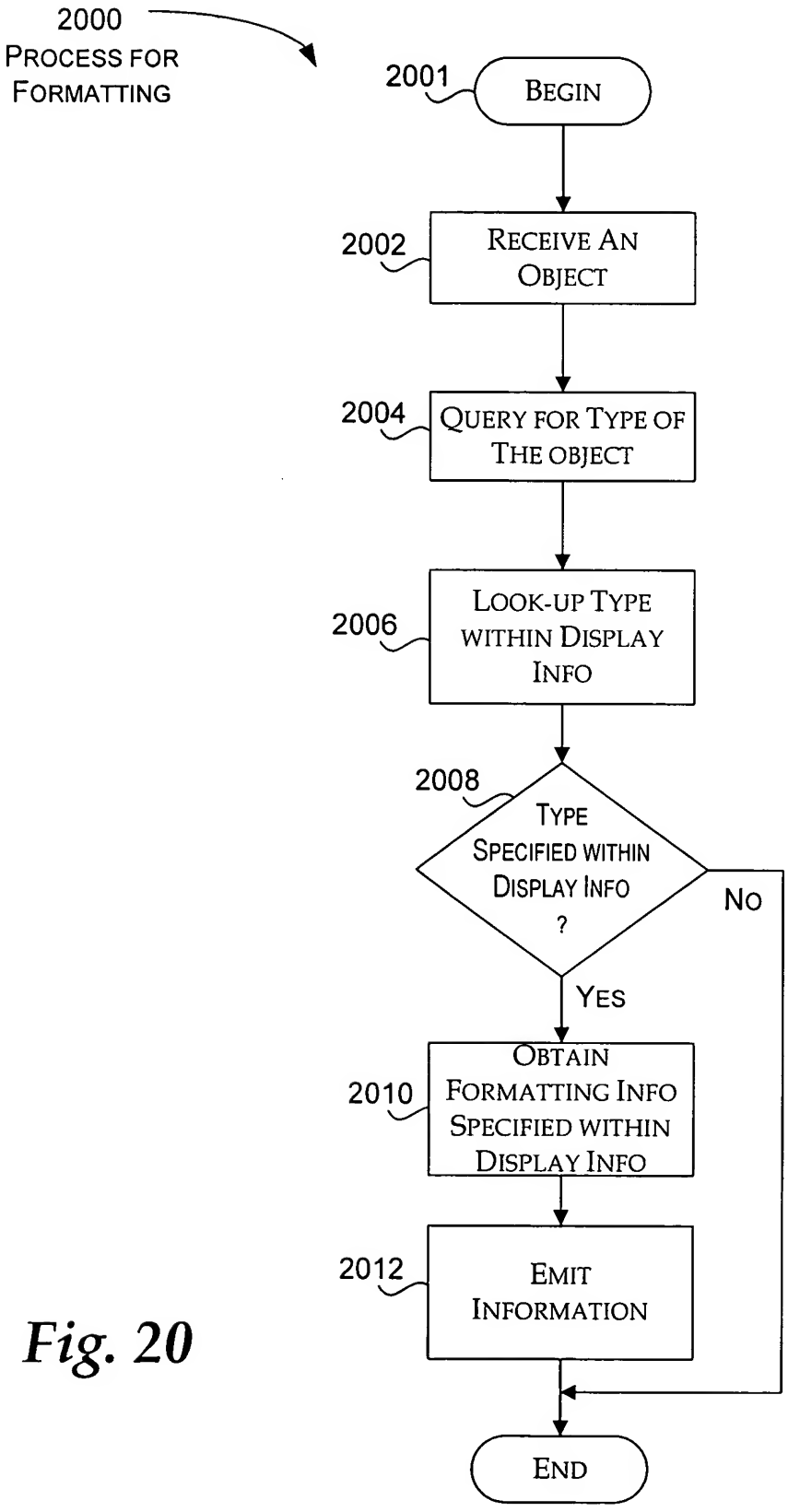
2000
PROCESS FOR FORMATTING
2001
BEGIN
2002
RECEIVE AN OBJECT
2004
QUERY FOR TYPE OF THE OBJECT
2006
LOOK-UP TYPE WITHIN DISPLAY INFO
2008
TYPE SPECIFIED WITHIN DISPLAY INFO ?
NO
YES
2010
OBTAIN FORMATTING INFO SPECIFIED WITHIN DISPLAY INFO
2012
EMIT INFORMATION
END

Fig. 20

2100
EXEMPLARY DISPLAY INFORMATION

```
<DISPLAYINFO>
 <TYPE> DIRECTORYINFO </TYPE>
 <SHAPE> TABLE </SHAPE>
 <COLUMNS>
  <COLUMN>
   <PROPERTYNAME> LASTWRITETIME </PROPERTYNAME>
  </COLUMN>
  <COLUMN>
   <PROPERTYNAME> SIZE </PROPERTYNAME>
   <ALIGN> LEFT </ALIGN>
   <WIDTH> 10 </WIDTH>
   <FORMATSTRING> D </FORMATSTRING>
   <LABEL> SIZE </LABEL>
  </COLUMN>
 </COLUMNS>
 <HEADER>
  ...
 </HEADER>
 <GROUPBY> PARENTPATH </GROUPBY>
 ...
</DISPLAYINFO>
```

Fig. 21

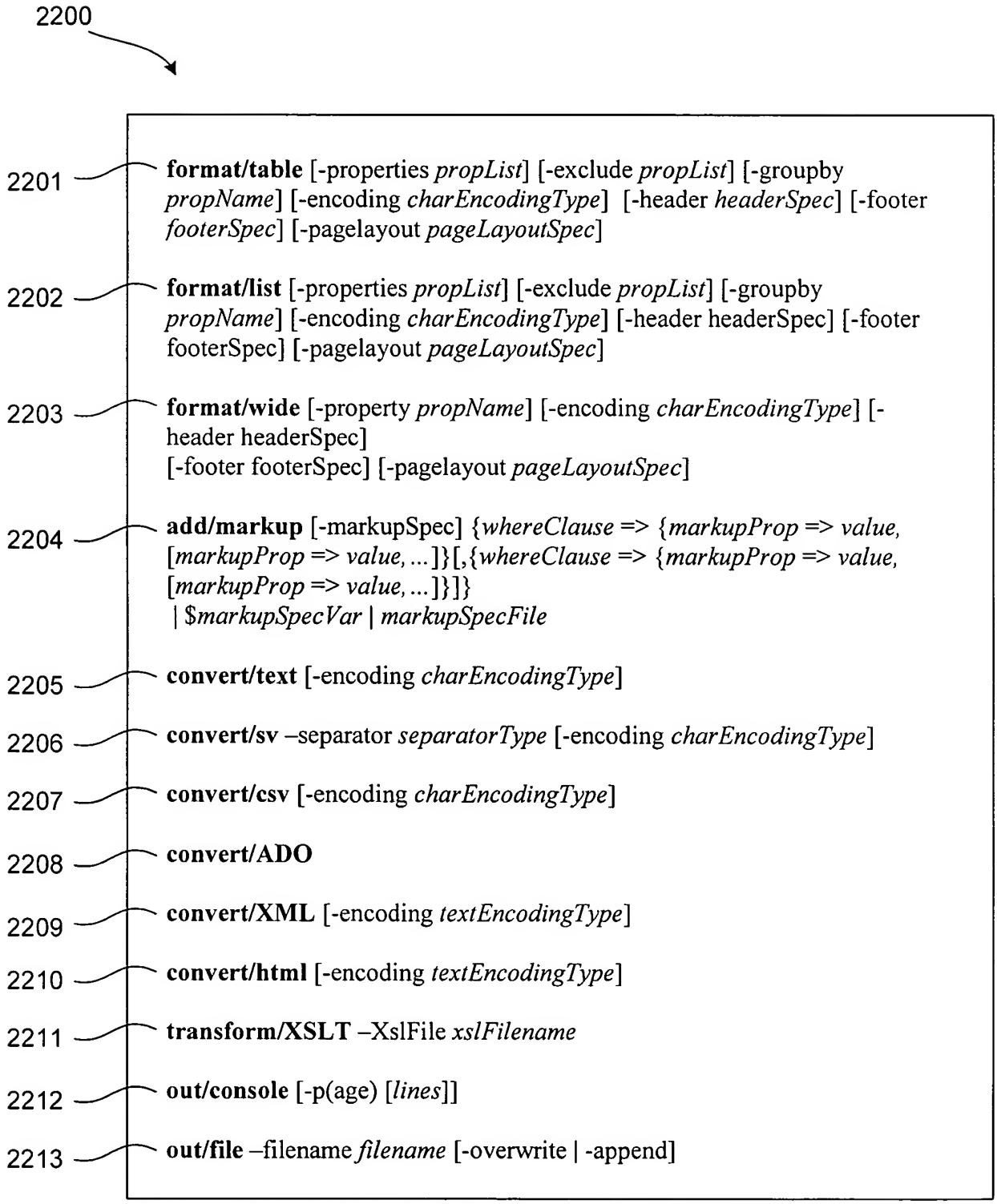
2200
2201
format/table [-properties propList] [-exclude propList] [-groupby propName] [-encoding charEncodingType] [-header headerSpec] [-footer footerSpec] [-pagelayout pageLayoutSpec]
2202
format/list [-properties propList] [-exclude propList] [-groupby propName] [-encoding charEncodingType] [-header headerSpec] [-footer footerSpec] [-pagelayout pageLayoutSpec]
2203
format/wide [-property propName] [-encoding charEncodingType] [-header headerSpec]
[-footer footerSpec] [-pagelayout pageLayoutSpec]
2204
add/markup [-markupSpec] {whereClause => {markupProp => value, [markupProp => value, ...]}[,{whereClause => {markupProp => value, [markupProp => value, ...]}]}
| $markupSpecVar | markupSpecFile
2205
convert/text [-encoding charEncodingType]
2206
convert/sv –separator separatorType [-encoding charEncodingType]
2207
convert/csv [-encoding charEncodingType]
2208
convert/ADO
2209
convert/XML [-encoding textEncodingType]
2210
convert/html [-encoding textEncodingType]
2211
transform/XSLT –XslFile xslFilename
2212
out/console [-p(age) [lines]]
2213
out/file –filename filename [-overwrite | -append]

Fig. 22

2300

```
[1]$ dir | format/table | out/console

Written                 Length.kb Name
6/28/2002 12:52:10 PM      189    sd.exe
11/7/2002 10:59:18 AM        0    sd.ini
6/28/2002 12:52:10 PM    2,633    sd.pdb
6/28/2002 12:52:18 PM      223    sdscc.dll

[2]$ dir | format/list brief.set | convert/text | out/console

Written   : 6/28/2002 12:52:10 PM
Length.kb : 189
Name      : sd.exe

Written   : 11/7/2002 10:59:18 AM
Length.kb : 0
Name      : sd.ini

Written   : 6/28/2002 12:52:10 PM
Length.kb : 2633
Name      : sd.pdb

[27]$ %mymarkupspec = {where length.kb -gt 500 => {applyto => all,
font.name => arial, font.style => bold, font.color => blue}
[28]$ dir i*| format/table | markup %mymarkupspec | out/console

Written                 Length.kb            Name
=================== ============ ========================
9/11/2002 3:11:10 PM       1,020          Interop.Excel.dll
9/11/2002 3:11:10 PM         192          Interop.Microsoft.Office.Core.dll
9/11/2002 3:11:10 PM         120           Interop.SHDocVw.dll
9/11/2002 3:11:09 PM          12           Interop.TASKSCHEDULERLib.dll
9/11/2002 3:11:10 PM          56           Interop.VBIDE.dll

[1]$ dir | format/table | convert/csv | out/console

"Written","Length.kb","Name"
"6/28/2002 12:52:10 PM","189","sd.exe"
"11/7/2002 10:59:18 AM","0","sd.ini"
"6/28/2002 12:52:10 PM","2633","sd.pdb"
"6/28/2002 12:52:18 PM","223","sdscc.dll"
```

Fig. 23